REQUEST DOCUMENT

Note:
This statement is available under UNIX, OpenVMS and Windows.

```
REQUEST DOCUMENT FROM operand1
[
 WITH
    [USER operand2]
    [PASSWORD operand2]
    [HEADER [[NAME] operand4 [VALUE] operand5]... ]
    [DATA { ALL operand6
          { [[NAME] operand7 [VALUE] operand8]... }]
]
{RETURN
    [HEADER [ALL operand9]
            [[NAME] operand10 [VALUE] operand11]...]

    [PAGE operand12]
}
RESPONSE operand13
[GIVING operand14]
```

Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
Operand1	C	S				A											no	no
Operand2	C	S				A											no	no
Operand3	C	S				A											no	no
Operand4	C	S				A											no	no
Operand5	C	S				A	N	P	I	F		D	T	L			no	no
Operand6	C	S				A	N	P	I	F	B	D	T	L			no	no
Operand7	C	S				A											no	no
Operand8	C	S				A	N	P	I	F		D	T	L			no	no
Operand9		S				A											no	yes
Operand10	C	S				A											no	yes
Operand11		S				A	N	P	I	F	B	D	T	L			no	yes
Operand12		S				A					B						no	yes
Operand13		S							I								no	yes
Operand14		S							I								no	no

Function

The REQUEST DOCUMENT statement gives you the means to access an external system.

Restrictions for Cookies

Under the HTTP Protocol, a server uses cookies to maintain state information on the client workstation.

REQUEST DOCUMENT is implemented using internet option settings. This means that, depending on the security settings, Cookies will be used.

If the internet option setting "Disabled" is set, no cookies will be sent, even if a cookie header (operand 4/5) is sent.

For Server environments, do not use the internet option setting "Prompt". This setting leads to a "hanging" server, because no client will be able to answer the prompt.

In the Windows environment, cookies are handled automatically by the Windows API. This means that, if cookies are enabled in the browser, all incoming cookies will be saved and sent automatically with the next request.

Note:
In a UNIX or OpenVMS environment, the following profile parameters have to be considered: NOPROX, PROXPORT, PROX.

For information on these parameters, refer to the profile parameter descriptions in the Natural Reference documentation.

operand1

Operand1 is the URL to access a document.

The information below is only valid if operand1 begins with "http://" or, in a Windows environment, also with "https://".

operand2

Operand2 is the name of the user that is used for the request.

operand3

Operand3 is the password of the user that is used for the request.

operand4/5

Operand4 is the name of a HEADER variable sent with this request.

Operand5 is the value of a HEADER variable sent with this request.

Note:
Operand4 and operand5 can only be used in conjunction with each other.

Header Name for Operand4

Header names are not allowed to contain CR/LF (carriage return/line feed) or ":" (colon). This will not be checked by the REQUEST DOCUMENT statement. For valid header names, please see the HTTP specifications. For compatibility with the web interface, header names can be written with "_" (underscore) instead of "-" (dash). (Internally, "_" is replaced by"-").

Header Value for Operand5

Header values are not allowed to contain CR/LF. This will not be checked by the REQUEST DOCUMENT statement. For valid header values and formats, please see the HTTP specifications.

General Information

For a HTTP request, some headers are required, eg: Request-Method or Content-Type.

These headers will be automatically generated depending on the parameters given with the REQUEST DOCUMENT statement.

Automatically Generated Headers (operand 4/5)

Request-Method

The following values are supported for operand5: "HEAD", "POST", "GET", and "PUT".

The following table shows the automatic calculation of Request-Method depending on the given operands:

Request-Method / Operand	HEAD	POST	GET	PUT
WITH HEADER **(operands 4/5)**	optional	optional	optional	optional
WITH DATA **(operands 7/8)**	not specified	specified	not specified	only with option ALL (operand 6)
RETURN HEADER **(operands 9 to 11)**	specified	optional	optional	optional
RETURN PAGE **(operand 12)**	not specified	specified	specified	optional

Content-Type

If the request method is POST, a content-type header has to be delivered with the HTTP request. If no content-type is set explicitly, the automatically generated value of operand5 is `"application/x-www-form-urlencoded"`.

Note:

It is possible to overwrite the automatically generated headers. Natural will not check them for errors. Unexpected errors may occur.

operand6

Operand6 is a complete document that is to be sent. This value is needed for the HTTP request method PUT.

operand7/8

Operand7 is the name of a DATA variable to be sent with this request. This value is needed for the HTTP request method POST (URL-encoding necessary, especially "&", "=", "%").

Operand8 is the value of a DATA variable to be sent with this request. This value is needed for HTTP request method POST (URL-encoding necessary, especially "&", "=", "%").

Note:
Operand7 and operand8 can only be used in conjunction with each other.

Restriction

If operand 7/8 is given, and the communication is "http://" or "https://" by default, the request method **POST** (see table above) with content type **application/x-www-form-urlencoded** is used.

During the request, the operands 7/8 will be separated by "=" and "&" characters. Therefore the operands are not allowed to contain "=", "&" and, because of URL-encoding, "%" characters. These characters are considered "unsafe" and need to be encoded as:

Character	URL-Encoding Syntax
%	%25
&	%26
=	%3D

General Note for URL-Encoding

When sending POST data with the content type **application/x-www-form-urlencoded**, certain characters must be represented by means of URL-encoding, which means substituting the character with %hexadecimal-character-code. The full details of when and why URL-encoding is necessary, are discussed in RFC 1630, RFC 1738 and RFC 1808. Some basic details are given here. All non-ASCII characters (i.e., valid ISO 8859/1 characters that are not also ASCII characters) must be URL-encoded, e.g., the file *köln.html* would appear in an URL as *k%F6ln.html*.

Some characters are considered to be "unsafe" when web pages are requested by e-mail.

These characters are:

Character	URL-Encoding Syntax
the tab character	%09
the space character	%20
[	%5B
\	%5C
]	%5D
^	%5E
‘	%60
{	%7B
|	%7C
}	%7D
~	%7E

When writing URLs, you should URL-encode these characters.

Some characters have special meanings in URLs, such as the colon (:) that separates the URL scheme from the rest of the URL, the double slash (//) that indicates that the URL conforms to the Common Internet Scheme syntax and the percent sign (%). Generally, when these characters appear as parts of file names, they must be URL-encoded to distinguish them from their special meaning in URLs (this is a simplification, read the RFCs for full details).

These characters are:

Character	URL-Encoding Syntax
"	%22
#	%23
%	%25
&	%26
+	%2B
,	%2C
/	%2F
:	%3A
<	%3C
=	%3D
>	%3E
?	%3F
@	%40

operand9

Operand9 contains all header values delivered with the HTTP response.

The first line contains the status information and all following lines contain the headers as pairs of name and value. The names always end in a colon (:) and the values end in a carriage return (CR). (Internally, all "CR/LF"s are transformed to "CR"s.)

operand10/11

Operand10 is the name of a HEADER received with this request. The HEADER is needed for HTTP.

Operand11 is the value of a HEADER received with this request. The HEADER is needed for HTTP.

Note:
Operand10 and operand11 can only be used in conjunction with each other.

Return Header Name for Operand10

For compatibility with the web interface, header names can be written with "_" instead of "-". Internally, "_" is replaced by "-".

If operand10 is a blank string, the status information is returned.

```
HTTP/1.0 200 OK
```

operand12

Operand12 is the document returned for this request.

operand13

Operand13 is the response number of the request (e.g. 200).

Overview of Response Numbers - for HTTP/HTTPs Requests

Status	**Value**	**Response**
STATUS CONTINUE	100	OK to continue with request
STATUS SWITCH_PROTOCOLS	101	Server has switched protocols in upgrade header
STATUS OK	200	Request completed
STATUS CREATED	201	Object created, reason = new URL
STATUS ACCEPTED	202	Async completion (TBS)
STATUS PARTIAL	203	Partial completion
STATUS NO_CONTENT	204	No info to return
STATUS RESET_CONTENT	205	Request completed, but clear form
STATUS PARTIAL_CONTENT	206	Partial GET fulfilled
STATUS AMBIGUOUS	300	Server could not decide what to return
STATUS MOVED	301	Object permanently moved
STATUS REDIRECT	302	Object temporarily moved
STATUS REDIRECT_METHOD	303	Redirection w/o new access method
STATUS NOT_MODIFIED	304	If-modified-since was not modified
STATUS USE_PROXY	305	Redirection to proxy, location header specifies proxy to use
STATUS REDIRECT_KEEP_VERB	307	HTTP/1.1: keep same verb
STATUS BAD_REQUEST	400	Invalid syntax
STATUS DENIED	401	Access denied
STATUS PAYMENT_REQ	402	Payment required
STATUS FORBIDDEN	403	Request forbidden
STATUS NOT_FOUND	404	Object not found
STATUS BAD_METHOD	405	Method is not allowed
STATUS NONE_ACCEPTABLE	406	No response acceptable to client found
STATUS PROXY_AUTH_REQ	407	Proxy authentication required
STATUS REQUEST_TIMEOUT	408	Server timed out waiting for request
STATUS CONFLICT	409	User should resubmit with more info
STATUS GONE	410	The resource is no longer available
STATUS LENGTH_REQUIRED	411	The server refused to accept request w/o a length
STATUS PRECOND_FAILED	412	Precondition given in request failed
STATUS REQUEST_TOO_LARGE	413	Request entity was too large
STATUS URL_TOO_LONG	414	Request URL too long
STATUS UNSUPPORTED_MEDIA	415	Unsupported media type
STATUS SERVER_ERROR	500	Internal server error
STATUS NOT_SUPPORTED	501	"Required" not supported
STATUS BAD_GATEWAY	502	Error response received from gateway
STATUS SERVICE_UNAVAIL	503	Temporarily overloaded
STATUS GATEWAY_TIMEOUT	504	Timed out waiting for gateway
STATUS VERSION_NOT_SUP	505	HTTP version not supported

Response 301 - 303 (Redirection)

Redirection means that the requested URL has moved. As a response, the Return Header with the name LOCATION will be displayed. This header contains the URL where the requested page has moved to. A new REQUEST DOCUMENT request can be used to retrieve the page moved.

HTTP browsers redirect automatically to the new URL, but the REQUEST DOCUMENT statement does not handle redirection automatically.

Response 401 (Denied)

The response "Access Denied" means that the requested page can only be accessed if a valid user ID and password are provided with the request. As a response, the Return Header with the name WWW-AUTHENTICATE will be delivered with the realm needed for this request.

HTTP browsers normally display a dialog with user ID and password, but with the REQUEST DOCUMENT statement, no dialog is displayed.

operand14

Operand14 contains the Natural error if the request could not be performed.

Examples

Note:

There is an example dialog V5-RDOC for this statement in the example library SYSEXV.

General Request

```
REQUEST DOCUMENT FROM "http://bolsap1:5555/invoke/sap.demo/handle_RFC_XML_POST"
  WITH
   USER #User PASSWORD #Password
   DATA
   NAME 'XMLData'       VALUE #Queryxml
   NAME 'repServerName' VALUE 'NT2'
  RETURN
   PAGE #Resultxml
  RESPONSE #rc
```

Simple Get Request (no data)

```
REQUEST DOCUMENT FROM "http://pcnatweb:8080"
  RETURN
   PAGE #Resultxml
  RESPONSE #rc
```

Simple Head Request (no return page)

```
REQUEST DOCUMENT FROM "http://pcnatweb"
   RESPONSE #rc
```

Simple Post Request (default)

```
REQUEST DOCUMENT FROM "http://pcnatweb/cgi-bin/nwwcgi.exe/sysweb/nat-env"
  WITH
   DATA
   NAME ’XMLData’       VALUE #Queryxml
   NAME ’repServerName’ VALUE ’NT2’
  RETURN
   PAGE #Resultxml
  RESPONSE #rc
```

Simple Put Request (with data all)

```
REQUEST DOCUMENT FROM "http://pcnatweb/test.txt"
  WITH
   DATA ALL  #document
  RETURN
   PAGE #Resultxml
  RESPONSE #rc
```